# Objetivos do Trabalho

O objetivo principal do trabalho foram:

- Estudar a estrutura e o funcionamento de Redes Neurais Convolucionais 2D
- Entender como a informação é transformada dentro da rede neural ao avançar nas camadas mais profundas
- Extrapolar esse conhecimento para redes mais clássicas como a **MLP** estudada durante a disciplina.

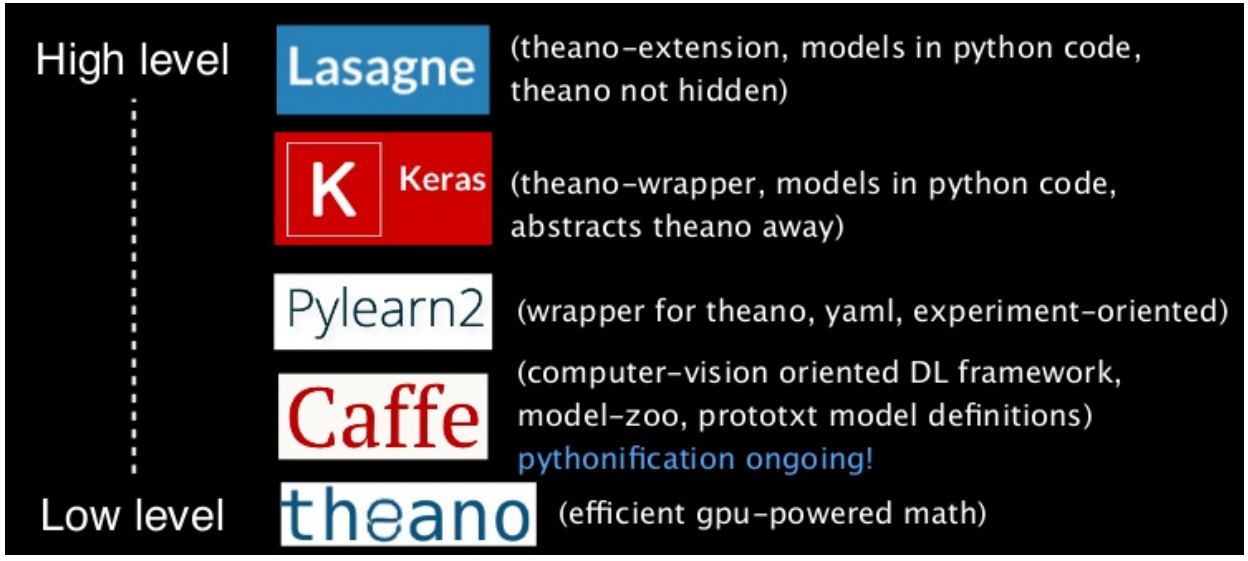
# Introdução e Motivação

# Apresentação do Bruno

# Apresentação do Bruno Canale

# Python - Keras Framework para Machine Learning

* Python - Linguagem de programação gratuita
* Contém uma quantidade muito grande de Frameworks voltados para *Machine Learning*




Keras foi inicialmente desenvolvido como parte de um projeto de pesquisa chamado de ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System)

# Keras - Exemplo de implementação MLP

```
modelo = Sequential()
modelo.add(Dense(num-neuronios, init='uniform'))
modelo.add(Activation('tanh'))

modelo.add(Dense(1, init='uniform'))
modelo.add(Activation('linear'))

modelo.compile(learning-rate=0.1, optimizer='sgd]')

modelo.fit(features-treino, target-treino)

modelo.predict(features-teste)
```

# Base de dados utilizada - MNIST

A base de dados MNIST é composta por 60.000 exemplos de imagens de digitos em letra cursivas. O dataset é ideal para testes de algoritmos em reconhecimentos de padrões por necessitar pouco pré-processamento.
Mais informações no link:
http://yann.lecun.com/exdb/mnist/

```
from keras.datasets import mnist

(X_treino, y_treino), (X_teste, y_teste) = mnist.load_data()
```

# Processamento necessário no atributo target

Um processamento padrão é transformar o conjunto de **atributos targets** em um conjunto de variáveis categóricas. O que seriam variáveis categoricas?
Exemplo: Se a lista de targets é composta por: [1.2, 2, 3, 4.2, 4i] e as classes disponiveis são **real, inteiro, imaginario**, então uma matriz de transformação seria:

Table: Conversão para variáveis categoricas

| target | real | inteiro | imaginario |
|---|---|---|---|
| 1.2 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 |
| 4.2 | 1 | 0 | 0 |
| 4i | 0 | 0 | 1 |

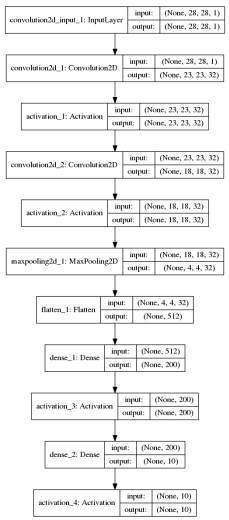
# Implementação da Rede Convolucional 2D

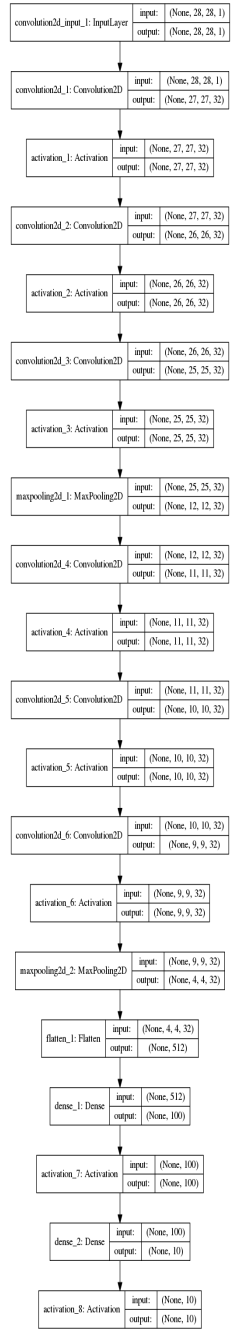
```
from keras.layers import Activation, Dense, Flatten
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D

num_filtros = 32
num_conv = 6
num_pool = 4

modelo = Sequential() # instaciar o modelo

modelo.add(Convolution2D(num_filtros, num_conv, num_conv,
                         border_mode='valid',
                         input_shape=(28, 28, 1)))
modelo.add(Activation('relu'))

# adicao da segunda camada convolucional
modelo.add(Convolution2D(num_filtros, num_conv, num_conv))
modelo.add(MaxPooling2D(pool_size=(num_pool, num_pool)))

# camada que transforma ('comprime') a saida em um array 1D
modelo.add(Flatten())

modelo.add(Dense(100, activation='relu'))
modelo.add(Dense(numero_classes_categoricas),
           activation='softmax')
```

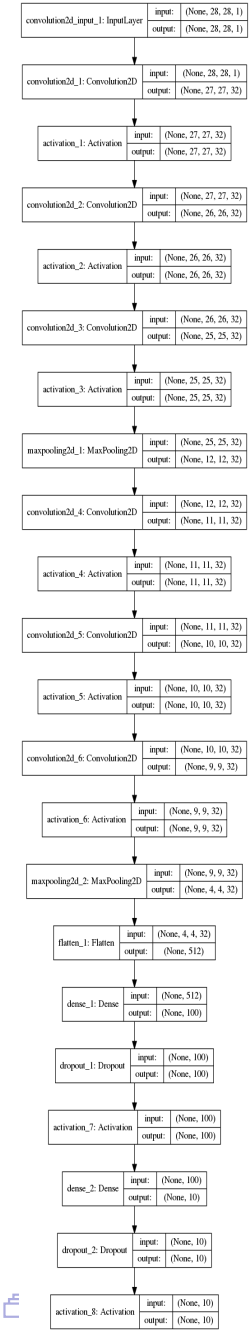
# Modelo do Experimento realizado para análise da Rede

- Após implementação a arquitetura foi testada para verificar performance no conjunto de testes.
- Queriamos observar a transformação da imagem de Input após cada camada da ConvNet
- A fim de analisar com mais detalhes, foram adicionadas mais camadas convolucionais no modelo apresentado anteriormente.

# Redes e Resultados

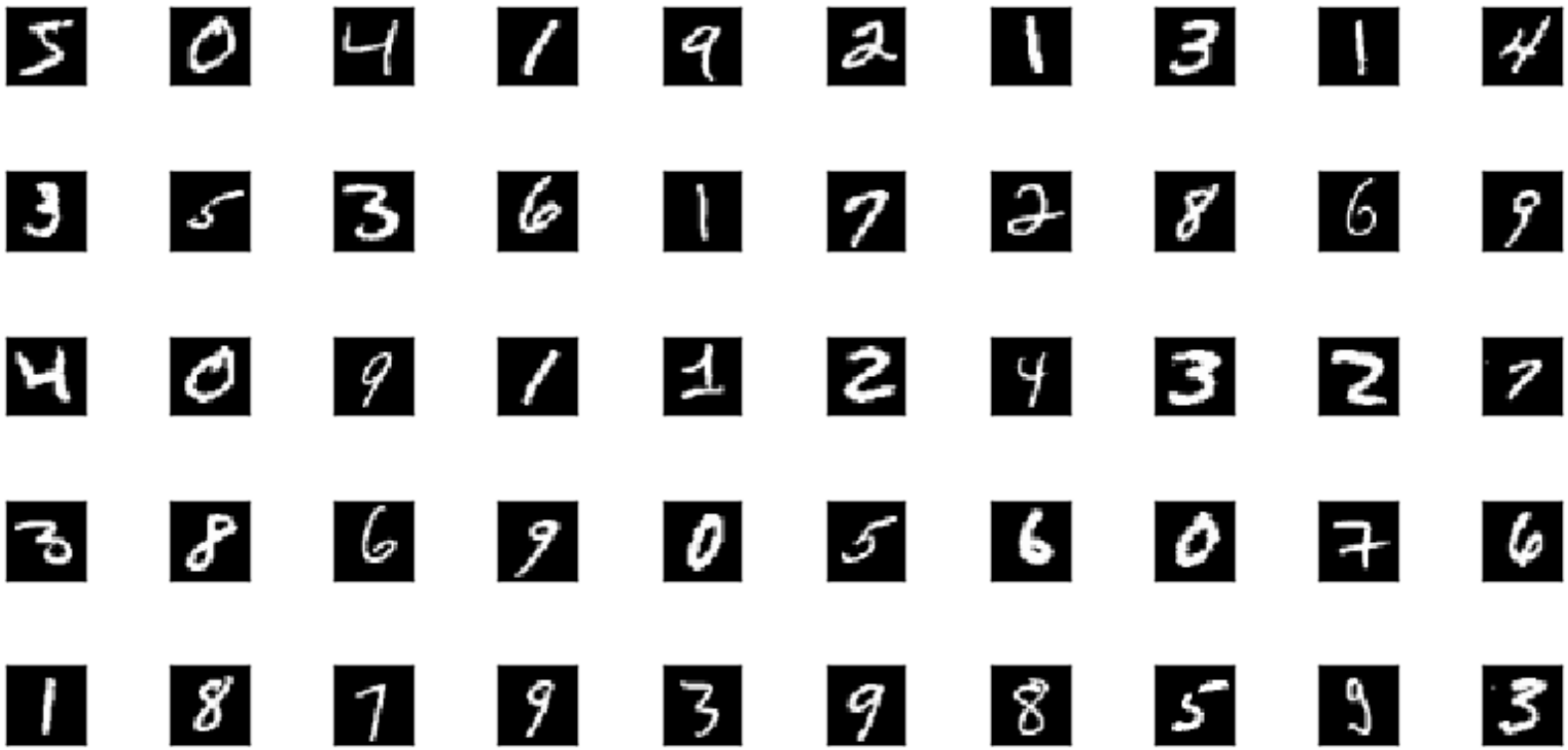


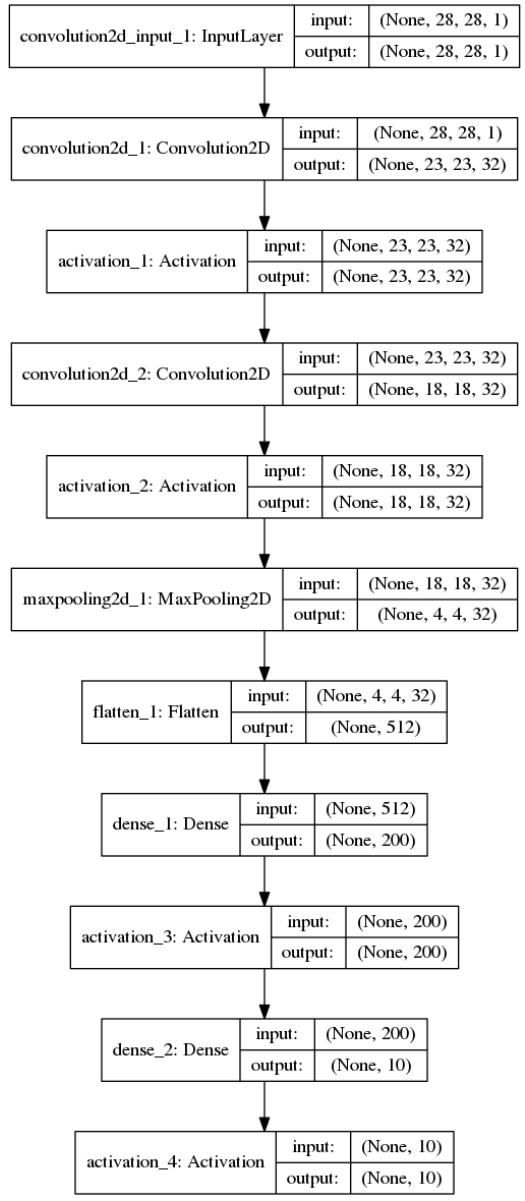



# Camada de entrada

MNIST DATASET



# Aplicação da CNN



# Treinamento

Treinamento

- Épocas = 10
- Itens = 60000
- Tempo = 30  40 minutos

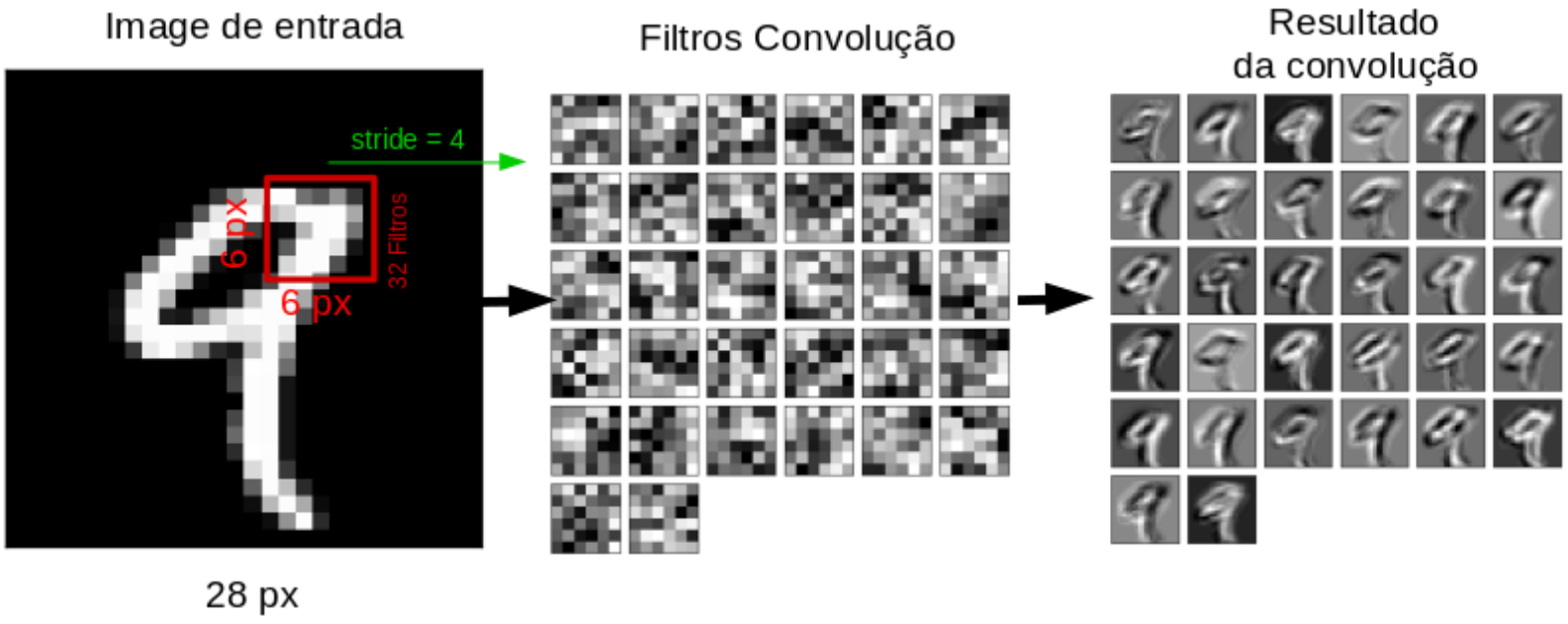
Teste

- Itens = 10000

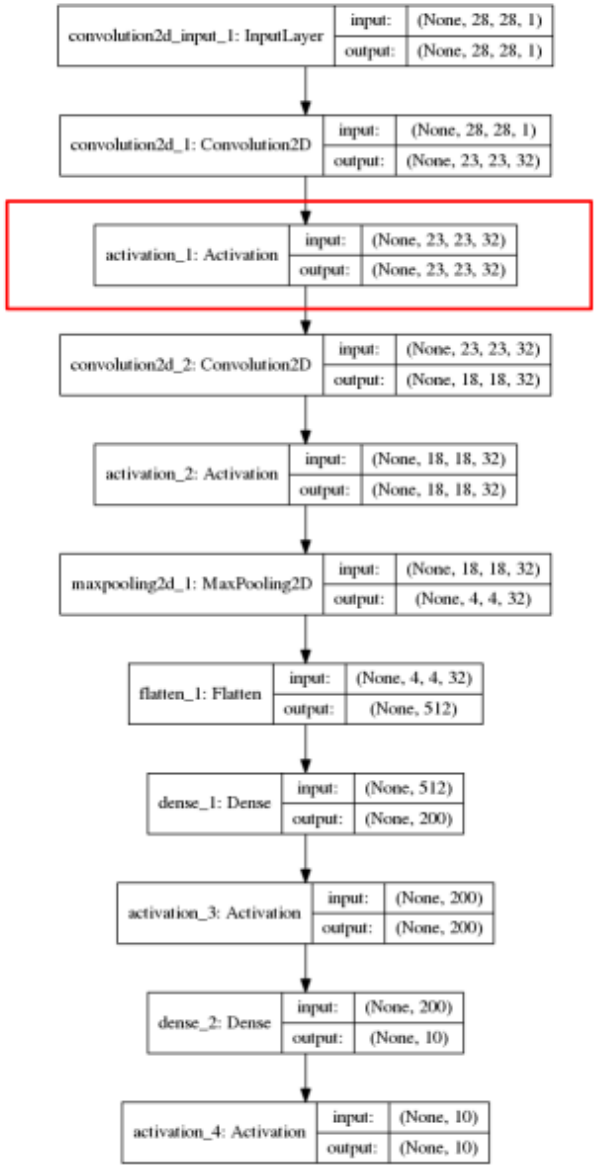
Resultado na base de teste

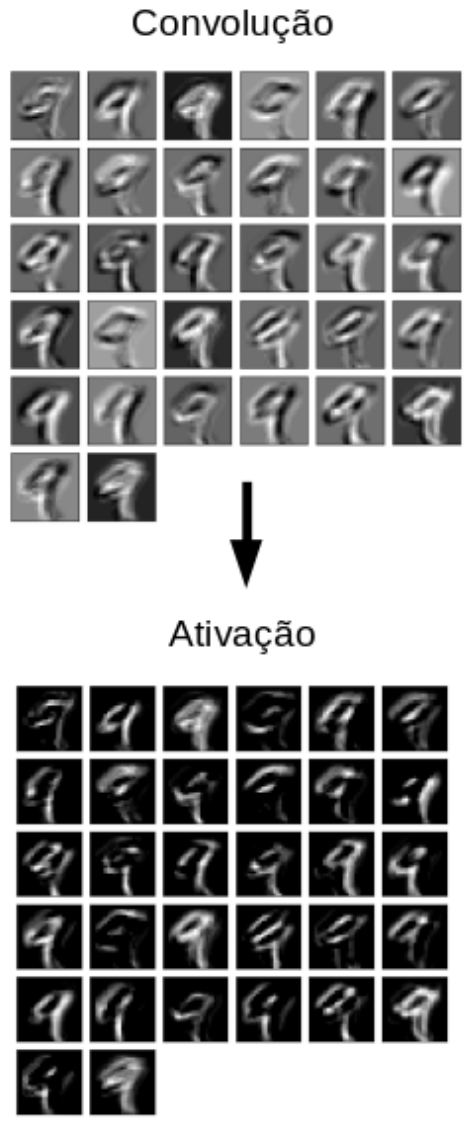
- 98.02%

# Convolução - 1

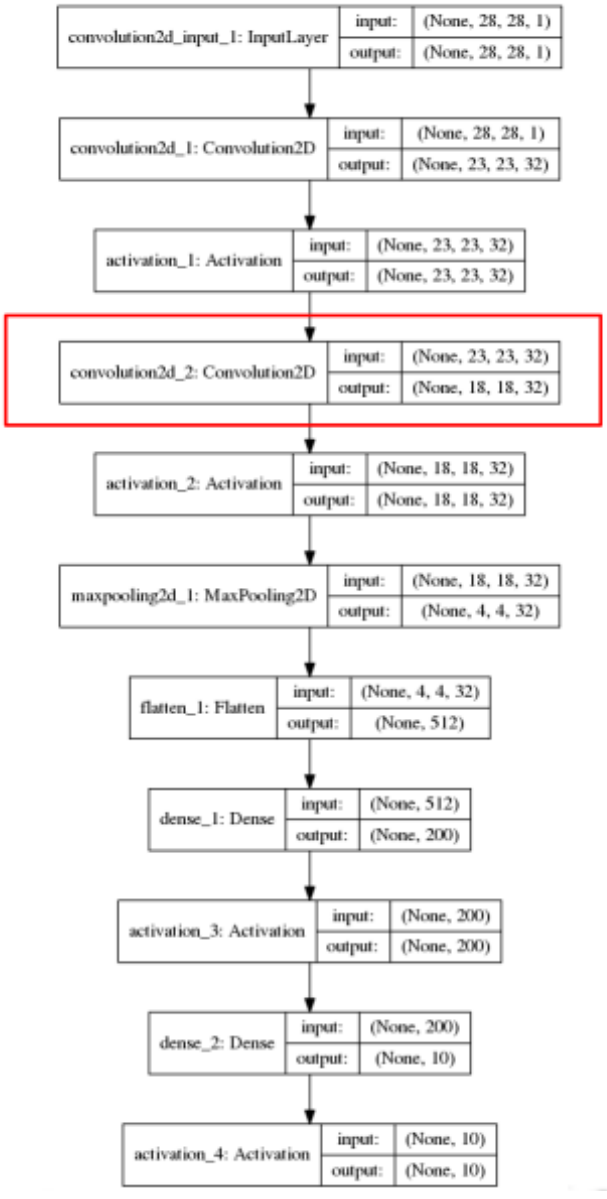


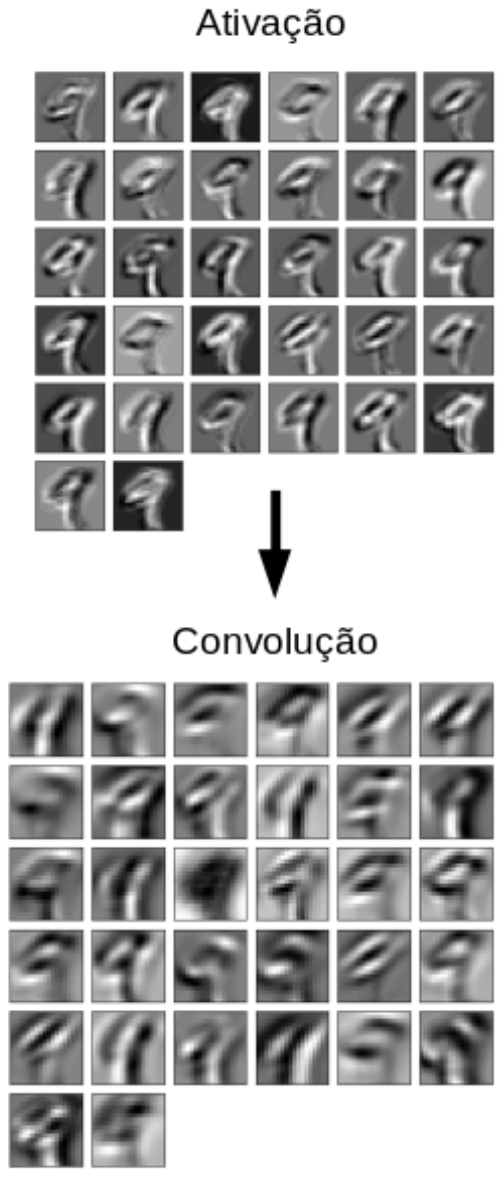
# Ativação - 1

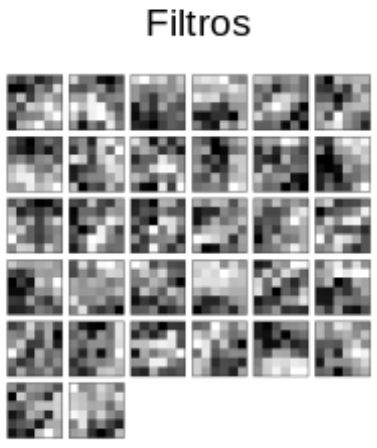

convolution2d_input_1: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
convolution2d_1: Convolution2D
input: (None, 28, 28, 1)
output: (None, 23, 23, 32)
activation_1: Activation
input: (None, 23, 23, 32)
output: (None, 23, 23, 32)
convolution2d_2: Convolution2D
input: (None, 23, 23, 32)
output: (None, 18, 18, 32)
activation_2: Activation
input: (None, 18, 18, 32)
output: (None, 18, 18, 32)
maxpooling2d_1: MaxPooling2D
input: (None, 18, 18, 32)
output: (None, 4, 4, 32)
flatten_1: Flatten
input: (None, 4, 4, 32)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 200)
activation_3: Activation
input: (None, 200)
output: (None, 200)
dense_2: Dense
input: (None, 200)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)


Convolução
Ativação

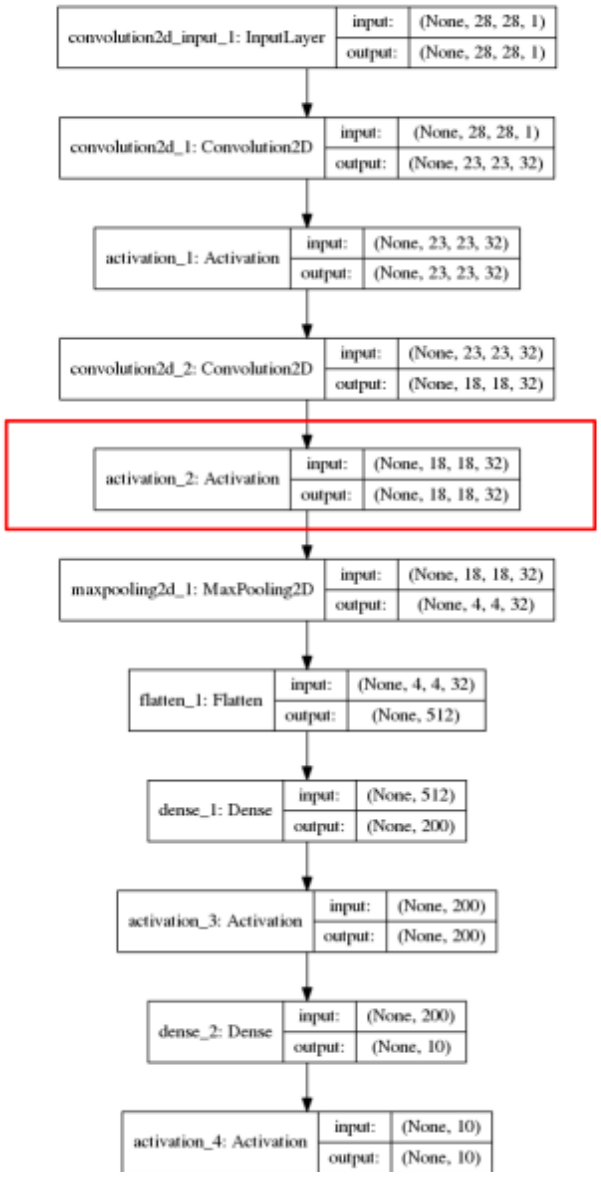
# Convolução - 2

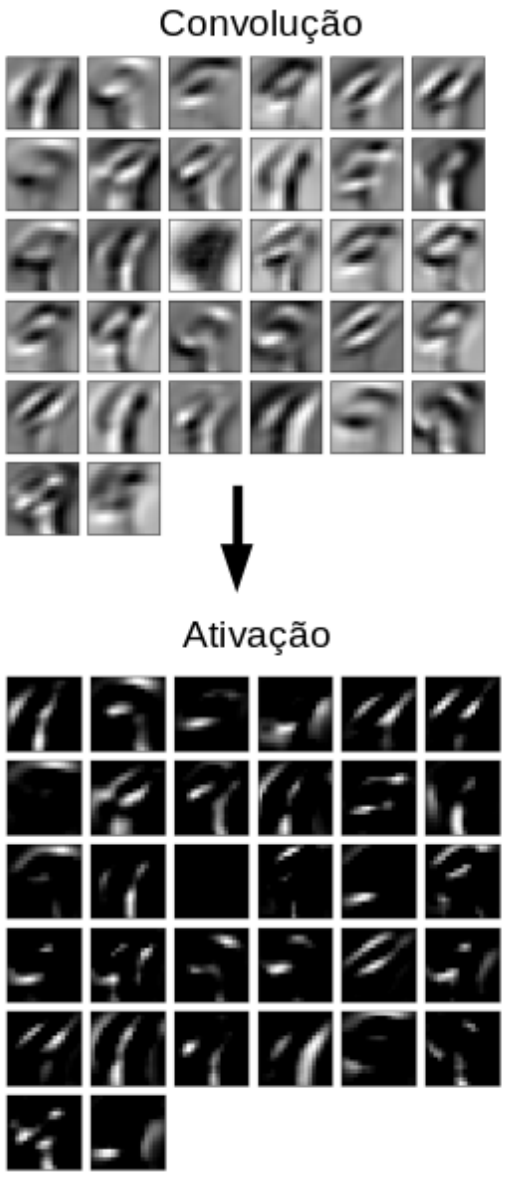


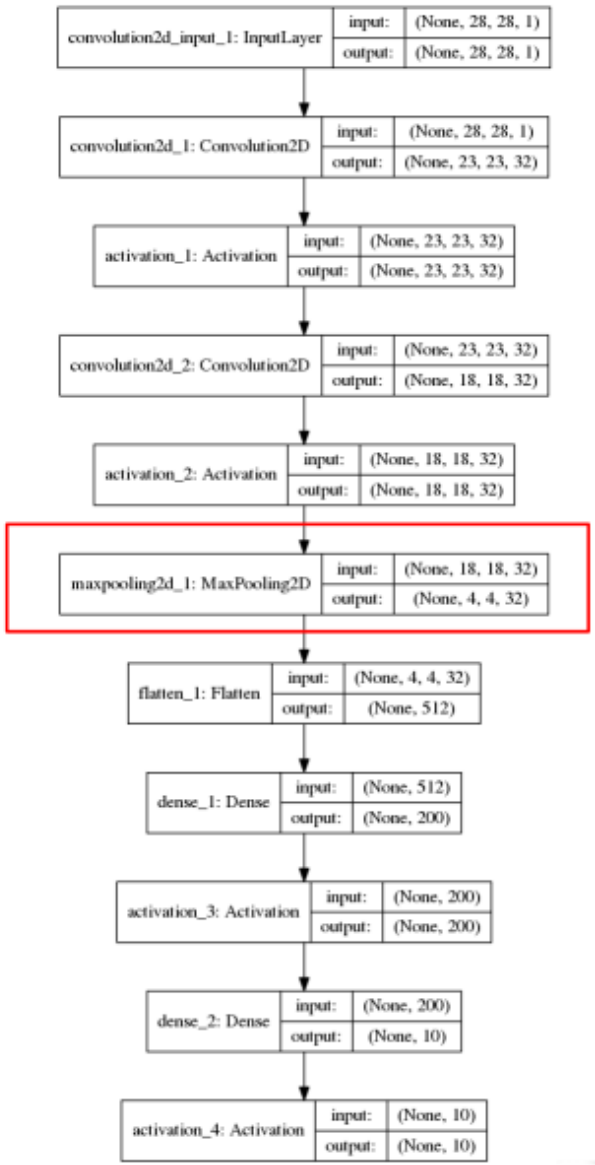



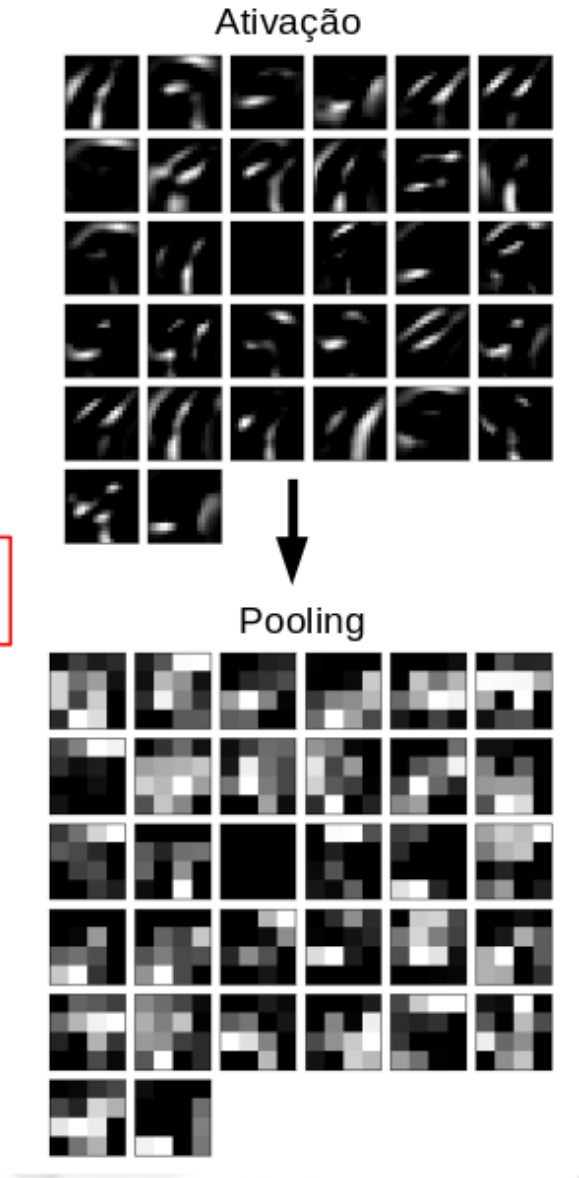
# Ativação - 2


convolution2d_input_1: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
convolution2d_1: Convolution2D
input: (None, 28, 28, 1)
output: (None, 23, 23, 32)
activation_1: Activation
input: (None, 23, 23, 32)
output: (None, 23, 23, 32)
convolution2d_2: Convolution2D
input: (None, 23, 23, 32)
output: (None, 18, 18, 32)
activation_2: Activation
input: (None, 18, 18, 32)
output: (None, 18, 18, 32)
maxpooling2d_1: MaxPooling2D
input: (None, 18, 18, 32)
output: (None, 4, 4, 32)
flatten_1: Flatten
input: (None, 4, 4, 32)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 200)
activation_3: Activation
input: (None, 200)
output: (None, 200)
dense_2: Dense
input: (None, 200)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)

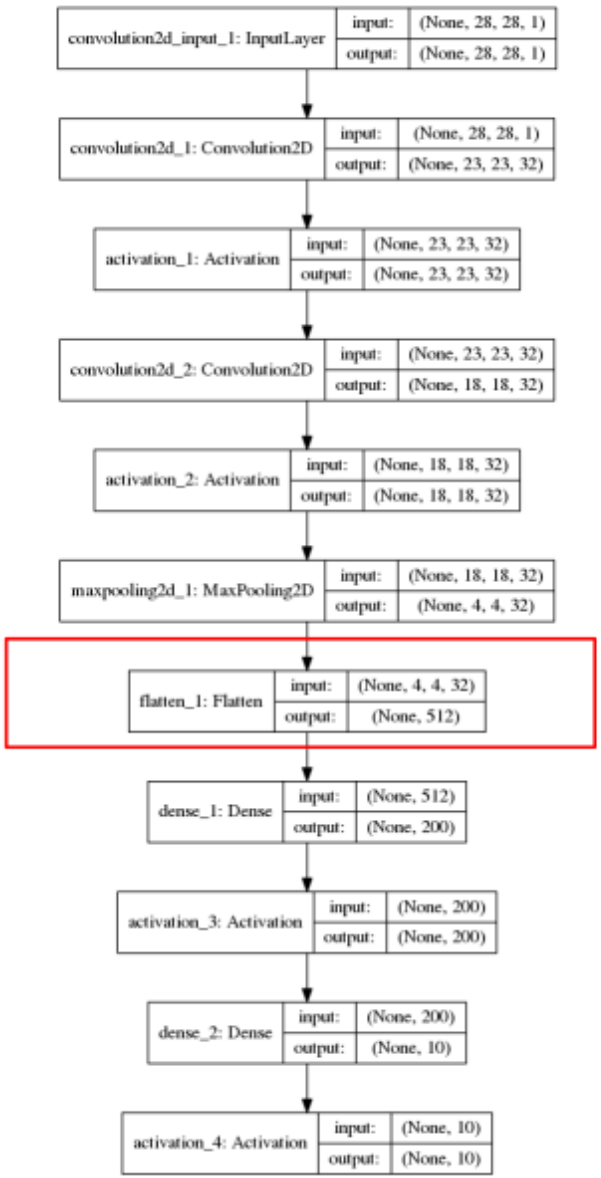

Convolução
Ativação

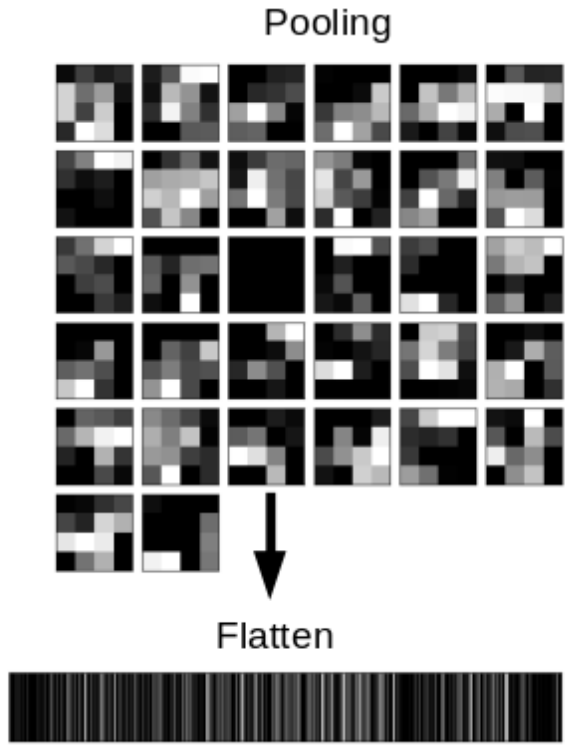
# Pooling


convolution2d_input_1: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
convolution2d_1: Convolution2D
input: (None, 28, 28, 1)
output: (None, 23, 23, 32)
activation_1: Activation
input: (None, 23, 23, 32)
output: (None, 23, 23, 32)
convolution2d_2: Convolution2D
input: (None, 23, 23, 32)
output: (None, 18, 18, 32)
activation_2: Activation
input: (None, 18, 18, 32)
output: (None, 18, 18, 32)
maxpooling2d_1: MaxPooling2D
input: (None, 18, 18, 32)
output: (None, 4, 4, 32)
flatten_1: Flatten
input: (None, 4, 4, 32)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 200)
activation_3: Activation
input: (None, 200)
output: (None, 200)
dense_2: Dense
input: (None, 200)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)

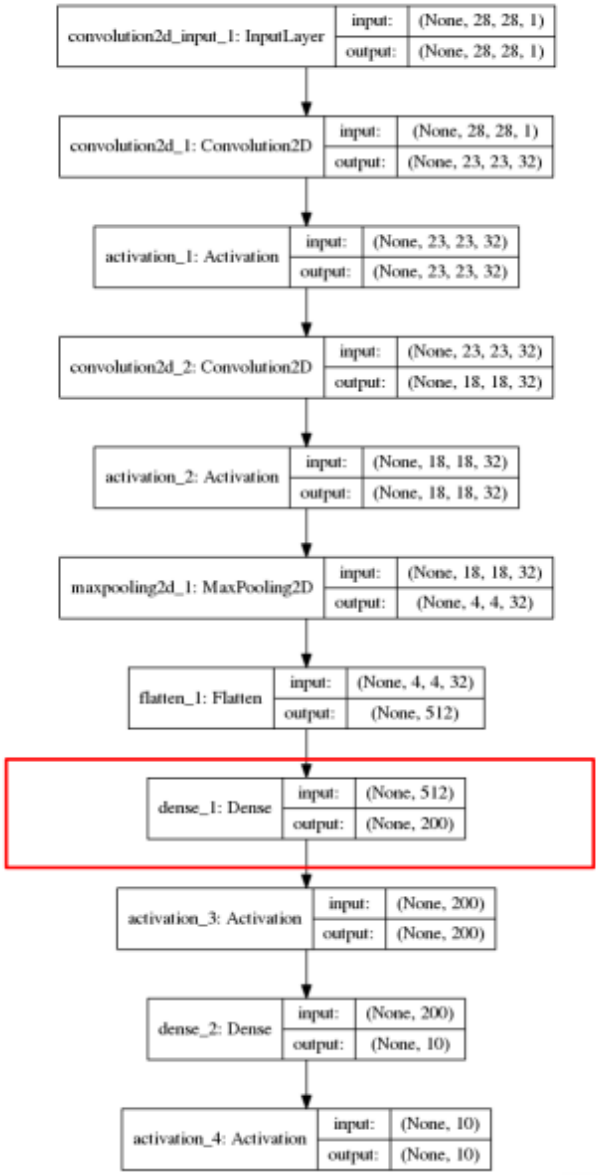

Ativação
Pooling

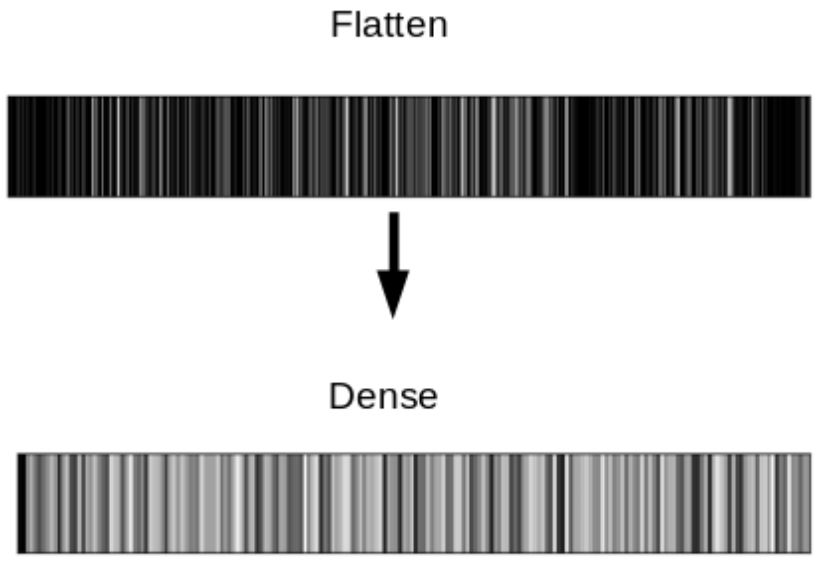
# Flatten ( N * 2D -> 1D)



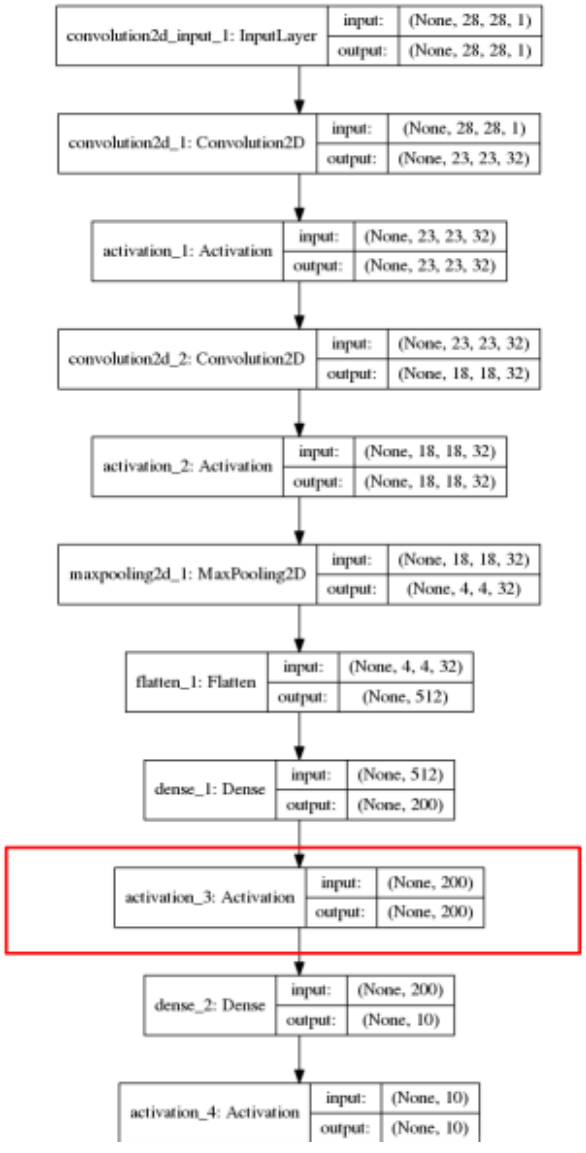



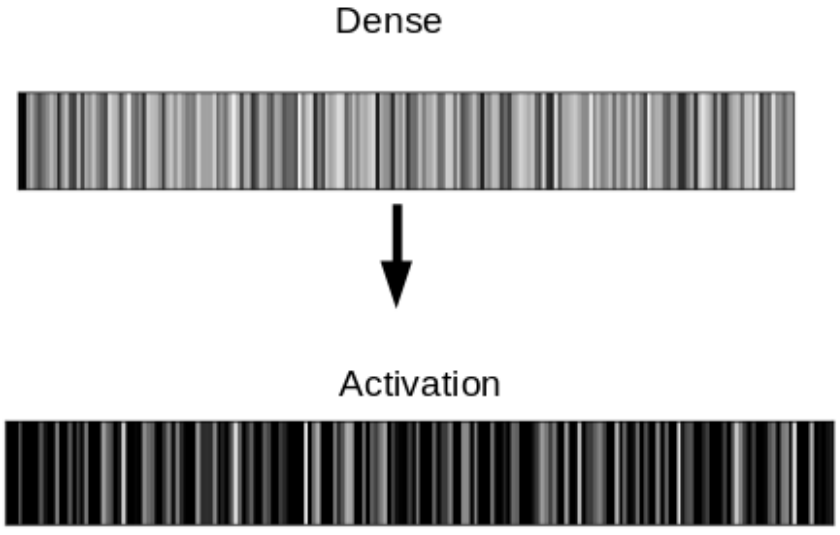
# Dense - 1


convolution2d_input_1: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
convolution2d_1: Convolution2D
input: (None, 28, 28, 1)
output: (None, 23, 23, 32)
activation_1: Activation
input: (None, 23, 23, 32)
output: (None, 23, 23, 32)
convolution2d_2: Convolution2D
input: (None, 23, 23, 32)
output: (None, 18, 18, 32)
activation_2: Activation
input: (None, 18, 18, 32)
output: (None, 18, 18, 32)
maxpooling2d_1: MaxPooling2D
input: (None, 18, 18, 32)
output: (None, 4, 4, 32)
flatten_1: Flatten
input: (None, 4, 4, 32)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 200)
activation_3: Activation
input: (None, 200)
output: (None, 200)
dense_2: Dense
input: (None, 200)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)

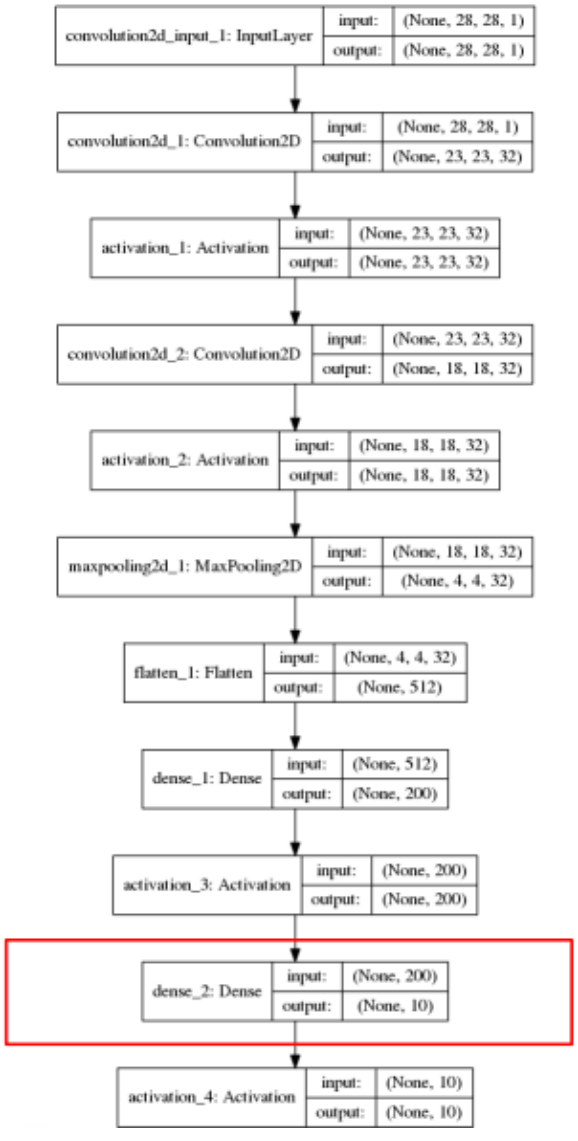

Flatten
Dense

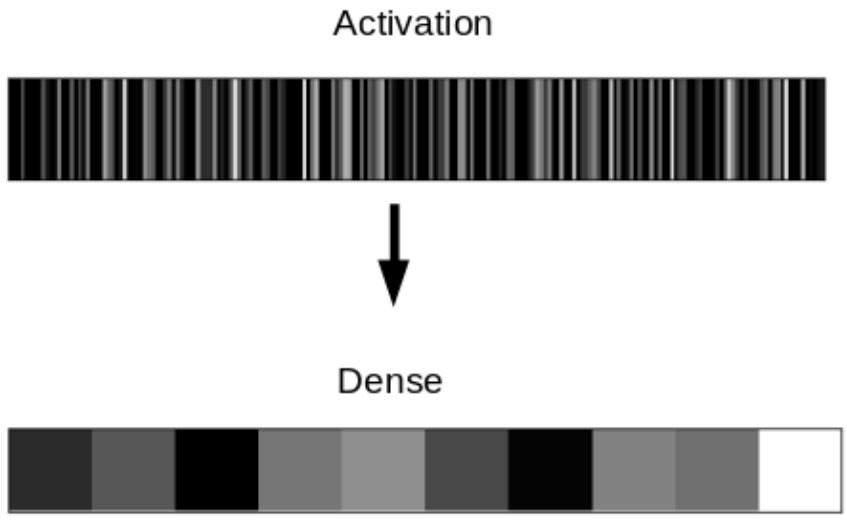
# Ativação - 3


convolution2d_input_1: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
convolution2d_1: Convolution2D
input: (None, 28, 28, 1)
output: (None, 23, 23, 32)
activation_1: Activation
input: (None, 23, 23, 32)
output: (None, 23, 23, 32)
convolution2d_2: Convolution2D
input: (None, 23, 23, 32)
output: (None, 18, 18, 32)
activation_2: Activation
input: (None, 18, 18, 32)
output: (None, 18, 18, 32)
maxpooling2d_1: MaxPooling2D
input: (None, 18, 18, 32)
output: (None, 4, 4, 32)
flatten_1: Flatten
input: (None, 4, 4, 32)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 200)
activation_3: Activation
input: (None, 200)
output: (None, 200)
dense_2: Dense
input: (None, 200)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)

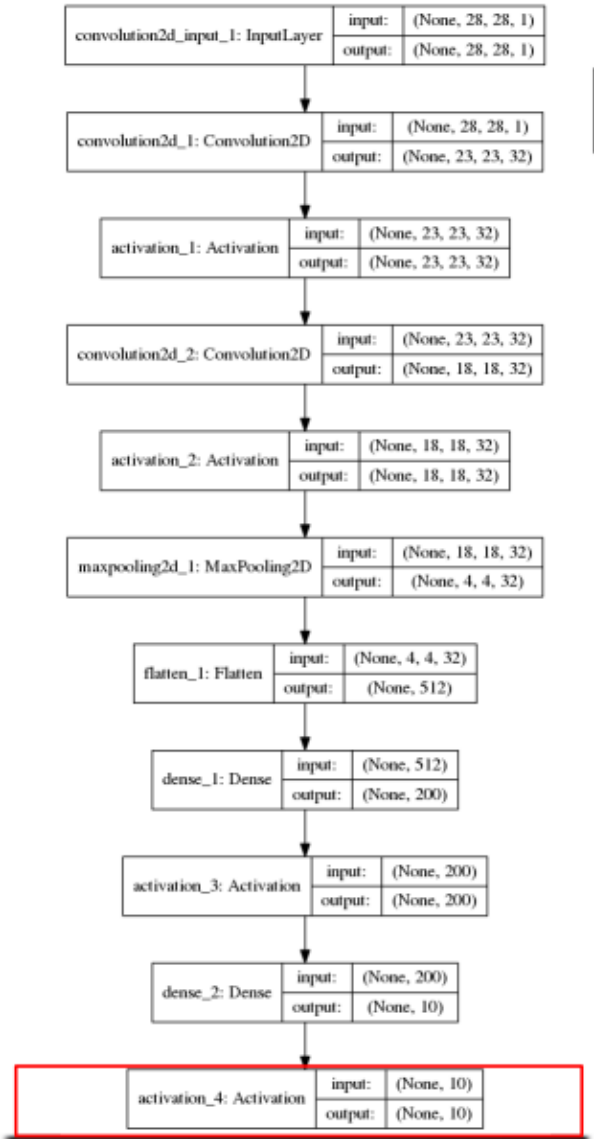

Dense
Activation

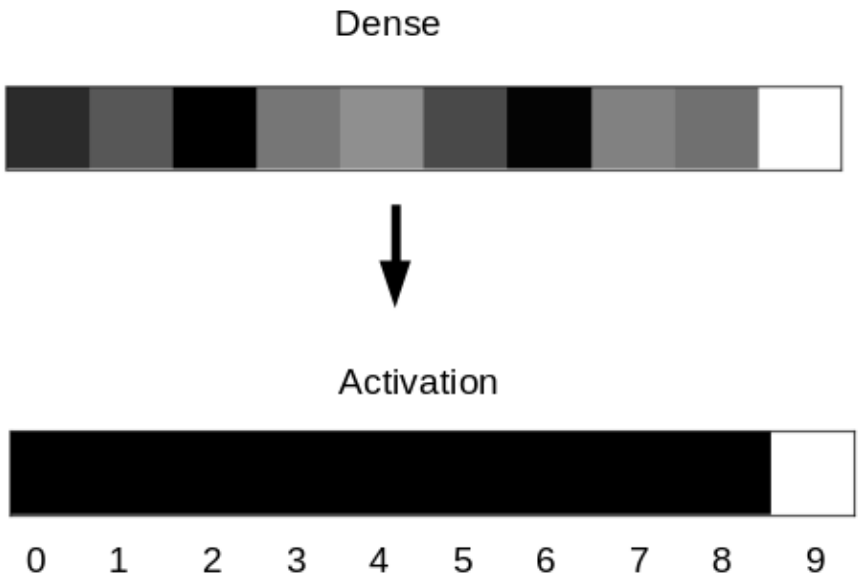
# Dense - 2


convolution2d_input_1: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
convolution2d_1: Convolution2D
input: (None, 28, 28, 1)
output: (None, 23, 23, 32)
activation_1: Activation
input: (None, 23, 23, 32)
output: (None, 23, 23, 32)
convolution2d_2: Convolution2D
input: (None, 23, 23, 32)
output: (None, 18, 18, 32)
activation_2: Activation
input: (None, 18, 18, 32)
output: (None, 18, 18, 32)
maxpooling2d_1: MaxPooling2D
input: (None, 18, 18, 32)
output: (None, 4, 4, 32)
flatten_1: Flatten
input: (None, 4, 4, 32)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 200)
activation_3: Activation
input: (None, 200)
output: (None, 200)
dense_2: Dense
input: (None, 200)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)


Activation
Dense

# Ativação - 4


convolution2d_input_1: InputLayer
input: (None, 28, 28, 1)
output: (None, 28, 28, 1)
convolution2d_1: Convolution2D
input: (None, 28, 28, 1)
output: (None, 23, 23, 32)
activation_1: Activation
input: (None, 23, 23, 32)
output: (None, 23, 23, 32)
convolution2d_2: Convolution2D
input: (None, 23, 23, 32)
output: (None, 18, 18, 32)
activation_2: Activation
input: (None, 18, 18, 32)
output: (None, 18, 18, 32)
maxpooling2d_1: MaxPooling2D
input: (None, 18, 18, 32)
output: (None, 4, 4, 32)
flatten_1: Flatten
input: (None, 4, 4, 32)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 200)
activation_3: Activation
input: (None, 200)
output: (None, 200)
dense_2: Dense
input: (None, 200)
output: (None, 10)
activation_4: Activation
input: (None, 10)
output: (None, 10)

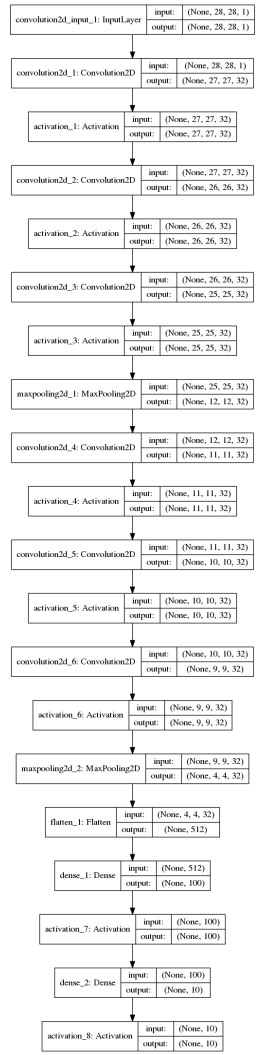

Dense
Activation
0 1 2 3 4 5 6 7 8 9

# Resultados das demais redes testadas - 1

Precisão na base de treino: 98.94%

Precisão na base de teste: 98.89%

3 conv + 1 pooling + 3 conv + 1 pooling + 2 FC

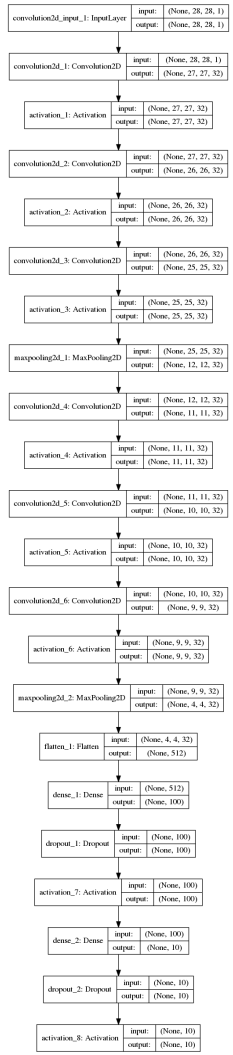


# Resultados das demais redes testadas - 2

Precisão na base de treino: 98.94%

Precisão na base de teste: 99.06%

3 conv + 1 pooling + 3 conv + 1 pooling + 2 FC *comdropout*



# MLP & CNN

- CNN é uma extensão do conceito da MLP
- Convoluções e Pooling ajudam a diminuir rapidamente o número de variáveis do sistema
- Próprio para o processamento de imagens e vídeos